STANDARD OPERATING PROCEDURE

FOR

ROUTINE OPERATION OF THE DATA ACQUISITION SYSTEM IN CRPAQS

STI-999214

Prepared By:
Mark Stoelting
Sonoma Technology, Inc.
1360 Redwood Way, Suite C
Petaluma, CA 94954-1169
(707) 665 - 9900
(707) 665 - 9800 fax

December 8, 1999

User Manual for the CRPAQS Data Acquisition System

DRAFT DOCUMENT: Some information is not in final form or is incomplete. This document has not been checked for spelling and format errors.
M. Stoelting, 8 November 1999.

The CRPAQS Data Acquisition System (DAS) is a flexible, programmable data collection system designed for air quality measurements. Data is collected on a standard, Pentium Personal Computer in standard ASCII files that can be read by any database parser, spreadsheet, or text editor. Both analog and serial data are collected, averaged, and stored during one-minute data collection intervals. Digital and serial commands operate pumps and valves and issue commands to conduct automated or manual calibration operations.

The systems were designed for maximum flexibility so that instrumentation can be easily added. All analog and serial input data are automatically recorded and archived at the site. Selected data are assigned database names and stored in files for FTP uploading to a remote server for daily, automated data processing and monitoring. All recorded data will be available on the DAS computer for diagnostic purposes and/or more detailed data analysis.

The DAS was initially designed for five CRPAQS monitoring sites at Fresno (FSF), Bakersfield (BAC), Angiola (ANG), San Jose (SJ4), and Sacramento (SDP). More sites will be added during the intensive sampling at the end of the program. Monitoring requirements at the sites are all slightly different, so the DAS hardware and data acquisition program setup varies accordingly.

Hardware

The DAS computer is a standard 500 MHz Pentium III enclosed in a rack-mount chassis. All systems have a 15-inch, high resolution monitor, a 56 Kbps modem, a 13 GB hard drive, a 1.4 MB floppy drive, an internal or external 100 MB Zip drive, and a CD-ROM drive. Sites expected to record large amounts of data (FSF, BAC, ANG) also have an 8 GB tape drive.

Data acquisition hardware installed in the DAS computers is supplied by National Instruments (NI) and Viewpoint Software. Analog inputs and digital input/output (DIO) functions at all monitoring sites are available through an NI DAQCard Model 6025E PCI adapter card. The card also provides analog output and counter/timer functions not currently utilized by the DAS. All systems include two built-in serial ports on the Pentium motherboard. Additional serial communications are provided through a Rocket Port 16-channel PCI adapter card from Viewpoint or a generic 2-port serial card.

The DAQCard provides sixteen 12-bit, single-ended, -10 to +10 volt, programmable gain analog input channels. For most signals, the resulting 4.88 mV measurement resolution

is adequate since 60 measurements are averaged for each datum. Very low voltage signals will require a reduction in the default input voltage range (increasing gain). The Fresno site has an AMUX 64-channel input multiplexer card expanding the analog input capability form 16 to 64 input channels. All input channel gain adjustments are done on the DAQCard, so each gain affects four mapped AMUX input channels. For example, a gain setting for channel 2 also affects AMUX channels 18, 34, and 50. Connections to analog inputs are done through the labeled, external AMUX board at the Fresno site.

At all the other sites, analog inputs signals are connected to the DAS through a simple screw terminal board with assignments listed in Table 1below.

Table 1. Screw terminal assignments for analog input channels at ANG, BAC, SJ4, and SDP.

Analog Input Channel	Screw terminal number	Analog input ground terminals
0	3	1 or 2
1	5	1 or 2
2	7	1 or 2
3	9	1 or 2
4	11	1 or 2
5	13	1 or 2
6	15	1 or 2
7	17	1 or 2
8	4	1 or 2
9	6	1 or 2
10	8	1 or 2
11	10	1 or 2
12	12	1 or 2
13	14	1 or 2
14	16	1 or 2
15	18	1 or 2

The DAQCard has 32 DIO lines that can be programmed to receive digital input signals or control the operation of pumps, flow valves, or other calibration related functions. On the DAS, only 8 or 16 of the DIO lines are currently used as control outputs. These lines operate interface relays that are electrically isolated from the DAS electronics. At the FSF, ANG, and BAC sites, sixteen control relays are available in an external ER-16 interface relay control box. Eight relay ER-8 boxes are installed at the SJ4 and SDP sites. The relays can switch up to 3A @ 120/240VAC suitable for controlling solenoid valves and other low power devices. Higher currents or highly inductive loads such as vacuum pumps should be switched via an interface relay using external power control relays such as part number 5X847 from WW Grainger (about $21). This power relay can control a 1½ HP motor, or 30A @ 120/240VAC.

As configured for annual sampling at the sites, all pumps and flow valves run on standard 120VAC power. At the ANG and BAC sites, 120VAC power is connected to the first 8 interface relay COM1 to COM8 terminals. The NO1control supplies switched power to a 5X847 power relay to run the calibrator pump. NO2 to NO6 supply switched power to air flow control valves. NO7 and NO8 controls have wiring connected but are unassigned.

The DAS uses serial ports or data collection and control. Two serial ports (COM1 and COM2) are available on all DAS installations through the two ports built in to the Pentium motherboards. At the FSF, ANG, and BAC sites, 16 additional serial ports (COM5 through COM20) are available through a Rocket Port PCI adapter card with an external connector chassis. At the SJ4 and SDP sites, a simple, two-port serial expansion card is installed as COM5 and COM6. Serial connections should be made to the Rocket Port hardware if possible because the adapter card handles serial communications more efficiently than the motherboard. All serial ports are capable of standard RS232 communications. The default port setting is 9600 baud, no parity, 8 data bits, 1 stop bit, and no flow control. These defaults may be superceded by DAS program configuration settings. At the ANG site where some instruments will be located on a tower, the Rocket Port hardware is capable of RS485 serial communications. This transmission specification is capable of much longer transmission distances and is less susceptible to interference.

Other computers will be located at the ANG and BAC sites for utility purposes, and to run the SMPS software. To facilitate file transmission between computers at these sites, a local 10 Mbps network is installed.

Software

The DAS computer runs under Windows NT 4.0, a multi-tasking, preemptive operating system capable of running several simultaneous tasks. The compute configuration is designed to automatically boot after any power interruption or any other non-destructive fault and resume normal data collection operations. The pcAnywhere remote control program runs as a background task. It monitors a phone line for incoming calls and provides password-protected access by off-site callers to the DAS file system and all DAS computer functions. The generic Windows NT scheduler also operates as a background task and monitors the computer clock. At specified time(s) each day using NT utilities, the scheduler initiates a call to an Internet Service Provider (ISP) and uploads data files to an FTP file server. As a foreground task, the CRPAQS Data Acquisition System program continuously collects and records data, performs automated calibrations, and displays current input values, strip charts, and control settings.

The DAS Windows NT operating system is version 4.0 Workstation with Service Pack 5 upgrade and is Y2K compliant. The computer BIOS is set to start when power is applied to allow automatic recovery from any power interruption. Programs required for DAS operation such as the data acquisition program and pcAnywhere remote access software

will execute automatically. As part of this automation, a default user name of "DAS" with default password of "crpaqs" automatically logs on to NT as the foreground user. This is required to provide real-time data displays and immediate access to the system by a local operator or a remote, dial-in user.

PcAnywhere is a commercial software product that provides remote access and control to the CRPAQS DAS computers. PcAnywhere is configured to automatically start operation when Windows is started. Depending on the type of communications available at the site, this software is configured for a modem dial-up access using a standard telephone line, or TCP/IP access through an internet connection. To be consistent across all the sites, DAS access is password protected with a standard User Name of "DAS" and a password of "crpaqs". Consult the pcAnywhere documentation for further details on program configuration and operation.

On a user-specified schedule, the DAS automatically uploads data files to any FTP file server on the internet. The upload schedule is set using the Windows NT "AT" scheduler utility bundled with the operating system. A task can be scheduled from the command line based using the example command, for example: AT 00:35:00 "UPLOAD.CMD".

If a dial-up internet connection is used, the DAS uses NT's Remote Access Services (RAS) to establish a connection and obtain an IP address. The specific RAS settings required for the connection are determined by the ISP. If an "on-demand" internet connection is available such as frame relay or ISDN, RAS dial-up services are not required. Once an internet connection established, the DAS uses NT's FTP comand utility to automatically upload data files to a server. An example UPLOAD.CMD file (including RAS dial-up) is:

```
rasdial [dial-up networking entry] [username] [password]
ftp -s:c:\ftp\send.txt ftp.sonomatech.com
rasdial /disconnect
```

An example send.txt command file would look something like:

```
DAS  (=username)
crpaqs   (=password)
cd stiin\crpaqs
put c:\ftp\fsfrecent.dat
quit
```

CRPAQS Data Acquisition System Program (DAP)

The CRPAQS Data Acquisition System program (DAP) controls all aspects of data collection, signal averaging, calibration control, and data storage. Many aspects of DAP operation are fixed, but the settings relating to data collection from instruments and control of calibration functions are fully programmable through specifications in the

CRPAQS.INI configuration file. The DAP file name is Crpaqs.exe written and compiled using the National Instruments LabView graphical programming language. The DAP program and matching configuration file is installed in the C:\Program Files\Crpaqs program directory. To automatically start the DAP when Windows starts, a Shortcut to Crpaqs.exe is located in the DAS user StartUp folder.

The CRPAQS Data Acquisition System monitor display panel is the primary source of real-time information all analog inputs, serial inputs, and control outputs and accepts user commands. A screen capture showing a display panel is shown in Figure 1. The upper left quadrant shows a large, scrollable table showing the details of all the DAS inputs. The lower right left shows the state of digital outputs used to control pumps, instruments, and control valves. The right part of the screen shows three graphs. The upper graph is always active and continuously variables selected by the operator. The lower two graphs may be activated by the user at any time, with a maximum duration of six hours after which the graph automatically turn off. When each of the two lower graphs is active, the DAP automatically stores one-second data files matching the graphed variables. For better screen resolution, a full-screen image of any graph may be viewed by using the Zoom button. Each graph has very powerful manipulation tools allowing users to change time scales and magnitudes, including the ability to blow up any portion of the graph to as large a scale as desired.

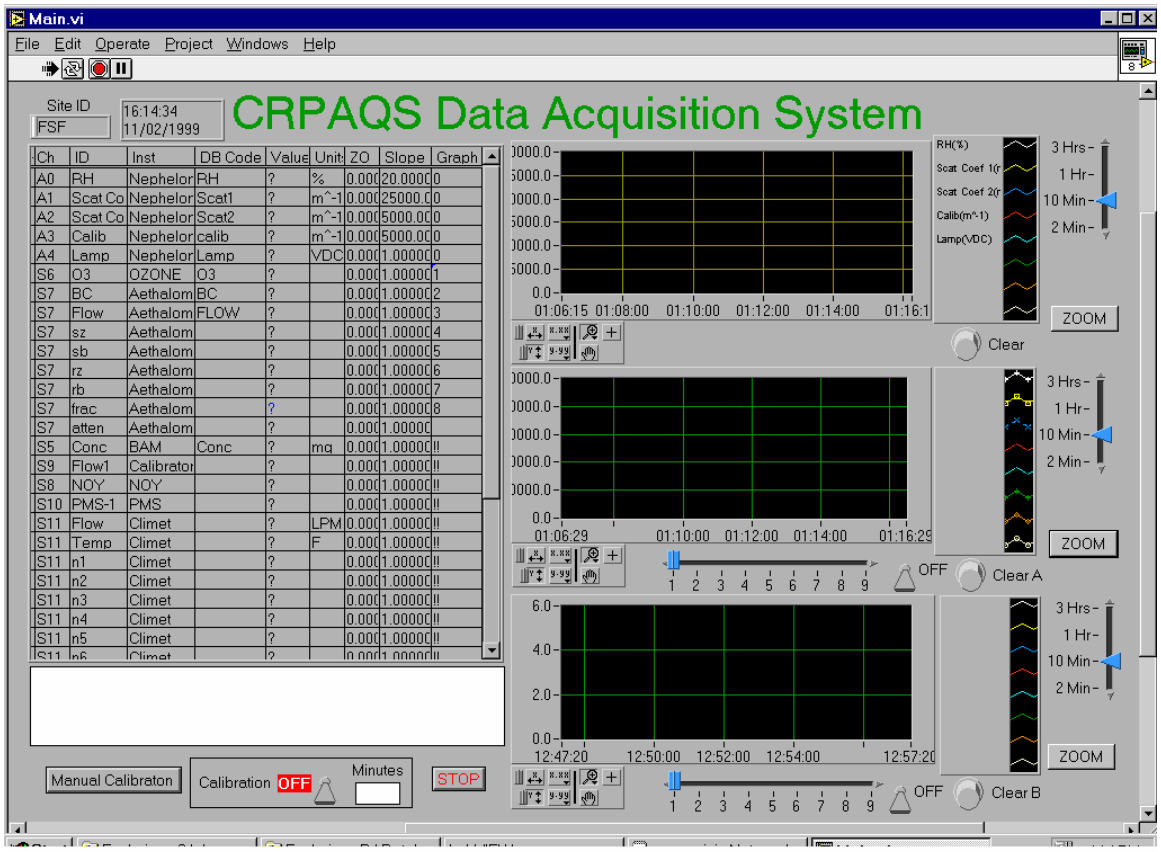

Figure 1. Screen capture of a typical DAS display panel.

The DAP operates on a one-second data acquisition cycle. Each second, all analog input voltages are measured, converted to engineering unit values and displayed on the screen. All serial port input buffers are emptied and the contents placed in their respective input serial data queues for processing, and immediately stored in a serial archive files. Each queue is checked to determine if a complete data record has been received. If so, the data record is parsed for specified data variables that are converted to engineering unit values and displayed on the screen.

Except for graph files, the DAP averages and stores all data on a one-minute time resolution. No information is carried forward to the next minute except an incomplete serial data record. Every minute, all analog input voltages are averaged and stored in an analog input archive file. The state of all control bits used to control pumps, valves, and instruments is also recorded in the analog input archive file. A subset of all analog voltages and parsed serial variables identified as the most important are converted to engineering units, averaged and recorded in two files. One file is a daily archive of these important data, and the second file is a continuously appended data file that is truncated every night at midnight. This second file (the RECENT file) is uploaded on a regular schedule to an FTP file server via the internet.

All data contained in the RECENT file are ingested by an automated database parser. As part of this process, each of these variables must have an assigned database ID (data_ID_n) that is also identified in the database. Database IDs are included in the RECENT file every day at midnight and every time the DAP starts recording data. Variables without an assigned database name are automatically left out of the RECENT file. RECENT variables are also automatically assigned an Op Code that contain information about the associated variable. Op Codes may indicate the presence of instrument warning flags, or indicated the recorded values are part of a calibration routine.

Depending on the DAP configuration specified in the Crpaqs.ini file, other operations also occur sporadically based on the one minute DAS time resolution. At the start of each minute, some serial instruments receive periodic commends to start acquiring data or send data. If a pre-defined calibration control sequence is in progress, the DAP checks the minute count and, if so instructed, sends serial commands to the calibrator or other serial instruments. As part of a calibration sequence, commands may be issued to change the state of pump, valves, or instruments using digital controls.

On a continuous basis, the DAP creates several archive data files and the RECENT data file used for uploading. At midnight every night, the DAP closes all data files for that day and truncates the RECENT data file to a specified number of data records. Then new archive files are created for the new day and the RECENT file is re-opened. As a special case, one-second containing graphed variables may span across a midnight boundary. Data directories and file names recorded by the DAP are indicated in the following table

where:

sss = three-letter site identifier
yyyy = year
mm = month
dd = day
pp = serial port number (COM port)
hhmm = graph file start time, file time stamp shows end time

Table 2. CRPAQS DAS data file locations and names.

Location	Name	Contents
D:\Data	sss RECENT.DAT	Most recent data for upload and database ingest
D:\Data\1M	sss yyyy mm dd.DAT	Daily 1-min RECENT archive
D:\Data\Analog	sss yyyy mm dd.MIN	Daily 1-min analog archive
D:\Data\Serial	sss yyyy mm dd pp.SER	Daily archive of each serial port
D:\Data\Graph	sss yyyy mm dd hhmm.SEC	1-sec graph file

DAP Configuration

All of the programmable features of the DAP except calibration command files are configured using the CRPAQS.INI initialization file. This self-documenting file is read by the DAP once upon starting, and immediately begins collecting data according to the instructions in the file. To make changes to DAP operations, edit CRPAQS.INI using Notepad, save the changes, and restart the DAP.

The .ini file has five main sections types:

1. [Configuration] sets the site ID, defines DAS hardware, identifies calibration file directory, and defines the truncation size of the "sss RECENT.DAT"
2. [AutoCal] sets the execution start time and specifies the .cal files used in regularly scheduled calibrations
3. [Instruments] identifies all the instruments connected to the DAS
4. [SERIALTEMPLATE] this section is used as a template when defining serial communication procedures with an instrument identified in the [Instruments] section
5. [ANALOGTEMPLATE] this section is used as a template when configuring analog input channels from an instrument identified in the [Instruments] section

At least one additional sections will be included one for each instrument identified in the [Instruments] section. Any instrument with both analog and serial DAS connections will have a separate .ini section for each.

An example CRPAQS.INI file is included as Table 3. This file was copied from a DAS that was collecting data from the instruments identified in the file.

Table 3. CRPAQS.INI configuration file.

```
; CRPAQS.INI
;
; NOTES:
; headers in bracket [] are called sections
; word before = is called the key, everything after = is called the value.
; Comments should be made on seperate lines because all characters after = and before
eol
; will be interpreted as part of the key's value.
; Blank lines are ok.
; All values are initally read as strings then converted to
; numbers if necessary. Quotes are not necessary. Be sure not to have extra spaces
; at the end of lines. They may be interpreted.
; Spaces are read as well so it is okay to uses spaces in filenames.
; If more than one instrument specifies the same Port number in Port= then the config
; used is the one specified LAST. During parsing, however, both will parse the data
; received in that port.
; Analog instruments specifing the same physical channel will read the same value.

[Configuration]
SiteID=FSF
; RecentFileSize is in 1 minute records
RecentFileSize=15000
; Analog input channels: if AMUX board is installed, use 16 channels even
; if 64 are available. Board should be installed using NIDAQ to have an
; AMUX board. When channel 0 is read, we get data for 0,1,2 and 3.
; In instrument definitions, go ahead and use channel 0-63
AnalogChannels=16
; digital inputs/output: specify 8 bit ports. 0,1 means bits 0-15. You cannot specify
; individual bits. 0,2 means bits 0-7 and 16-23. 1 means 8-15. Do not use
; colon (:) seperators. CRPAQS parses on commas
DigitalInputPorts=
DigitalOutputPorts=2,3
; com ports are 1 based com_1 = windows com1
; use com_n=1 to use standard windows com port
; use com_n=2 to use multicom serial port
; use com_n=3 to simulate data from a file name "Simulate Com n.txt"
; use com_n=0 (or leave out of list) to disable a port
; SerialPorts is the total number of ports on the computer
SerialPorts=20
com_1=0
com_2=0
com_3=0
com_4=0
```

```
com_5=2
com_6=2
com_7=2
com_8=2
com_9=2
com_10=2
com_11=2
com_12=2
com_13=2
com_14=0
com_15=0
com_16=0
com_17=0
com_18=0
com_19=0
com_20=0
; Directories are absolute from the root.
CalScriptDir=D:\calibrate

; AutoCal scripts. There can be any number of autocal scripts but they
; must be sequential (ie. don't have script1 then script3).
; StartTime should be specified in HH:MM in 24 hour time.
; If the specified file can not be found, nothing will happen. No error
; will occur it will simply be ignored.
[AutoCal]
Script1="AutoCal A.cal"
StartTime1=11:30
Every_nth_day1=1
Script2="AutoCal B.cal"
StartTime2=03:45
Script3="AutoCal C.cal"
StartTime3=04:45

; Instruments: can have any number of instruments. If an instrument has analog and serial
; data points, define 2 instrument. EX: [NOX_ANALOG] and [NOX_SERIAL]
[Instruments]
Instrument1=Nephelometer
Instrument2=OZONE
Instrument3=PMS
Instrument4=BAM
Instrument5=NOY
Instrument6=Calibrator
Instrument7=OCEC
Instrument8=Aethalometer
Instrument9=Climet
```

```
[SERIALTEMPLATE]
; Serial -- non-zero value means it is a serial instrument.
Serial=1
; ports are 1 based com Port=1 means Com1
Port=1
;
; OP code definitions
; if IO_control AND OP_Code_Mask = OP_code_x then OP_code = x
; where IO_control is the 32 bits represent all the digital IO
; and where OP_code is the number used to mark the data
; and where OP_Code_Mask and OP_Code_x are defined below.
; (values below are 32 bit binary numbers, lsb to the left, spaces ignored)
; Max of 9 op codes. missing opcodes will be ignored.
OP_Code_Mask=0100 0100 0000 0000
OP_Code_1=0100 0100 0000 0000
; ------------------- serial port configuration --------------------
; if left blank or not existant, 9600,n,8,1 is used.
BaudRate=
DataBits=
; StopBits: 0=1 bit, 1=1.5, 2=2 bits
StopBits=
; Parity: 0=no parity, 1=odd, 2=even
Parity=
BufferSize=
; XONXOFF: 0 for off or 1 for on
XONXOFF=
; Handshake: 0=none, 1=hardware
Handshake=

; Init_string is sent to instrument at startup
Init_string="Let's get started.\0D\0A"
; Recur_Cmd is a string sent periodically to the instrument. If left out,
; nothing is sent.
; Recur_Freq is the time, in minutes, between commands. If left out but
; Recur_Cmd is present and non-null string, Recur_Freq=1
; ALL NON PRINTABLES MUST BE ENTERED USING \xx where xx is the hex ascii
; carriage return = \0D   linefeed = \0A
Recur_Cmd="Send me data!!!\0D\0A"
Recur_Freq=1

; parse_delimiters: each character will be used to divide fields in the
; input string. use \xx for non printables.
parse_delimiters=", :\0A="
parse_endofrecord="\0A"
; Record is size is used only if no Parse_endofrecord is given. If no
; parse_recordsize is provided or it is <=0, default of 80 is used.
```

```
parse_recordsize=70

; data point information.
; data_count - how many data points for this instrument. Tells parser
; how many data_ID_n keys to look for. If missing, parser will look for
; data_ID_n until none is found.
data_count=3
; if no data_ID is given or no matchpattern, data will not be parsed
data_ID_1=NOX
; if database_name is not given or empty, data will not be stored in
; data files but if ID and matchpattern are given, data will be displayed
data_database_name_1=NOX
;Regular expressions are ok in matchpattern (see notes at bottom of file)
; note that extra spaces at the end of the line will be interpreted as part
; of the string. Use \\ before special regular expression characters to cancel
; the effect. ex: use \\* to match * but use just * to use it as regular expression
data_matchpattern_1=nox=
; data_offset_n: number of fields after end of pattern to find value (can be neg), optional
; if 0, searches match field for data, if -1 searches field before match.
data_offset_1=1
; data_char_offset_n is number of character past beginning of selected field (neg ok)
; fields begin after a delimiter.
; ex: "NOX=3.25"  matchpattern=NOX  offset=0  char_offset=0  delimiters include '='
; engineering unit value=(volts-zero_offset)*slope
data_char_offset_1=0
; slope and offset are the same as for analog data points
data_zero_offset_1=0.0
data_slope_1=1.0

data_ID_2=Inst_Status
data_matchpattern_2="Status="
; data_OP_Code_value. Data points can be used strictly as instrument state
; checkers. If an instrument sends, for instance, a string like
; "Status=----" we can define a data point that searches for "Status="
; then compares the next field ("----") with the string defined by
; data_OP_Code_value_n. If the field does not match, the value in
; data_OP_Code_n will be the opcode for data from that instrument.
data_OP_Code_value_2="----"
data_OP_Code_2=5

data_ID_3=Gasoline
data_database_name_3=gas
data_matchpattern_3=gas=
data_offset_3=1
```

```
[ANALOGTEMPLATE]
Serial=0
data_count=5
; ----------first analog data point----------
; Channel_n is physical input channel
; zero_offset_n default is 0.0
; slope_n default is 1.0
; min_n is -10.0V
; max_n is +10.0V
data_ID_1=RH
data_database_name_1=RH
data_Channel_1=0
data_zero_offset_1=0.0
data_slope_1=20.0
data_min_1=0
data_max_1=5
data_units_1="%"
; ------------------------------------------------
data_ID_2=Scat Coef 1
data_database_name_2=Scat1
data_Channel_2=1
data_zero_offset_2=0
data_slope_2=25000
data_min_2=0
data_max_2=0.5
data_units_2="m^-1"
; ------------------------------------------------
data_ID_3=Scat Coef 2
data_database_name_3=Scat2
data_Channel_3=2
data_zero_offset_3=0
data_slope_3=5000
data_min_3=0
data_max_3=5
data_units_3="m^-1"
; ------------------------------------------------
data_ID_4=Calib
data_database_name_4=calib
data_Channel_4=3
data_zero_offset_4=0
data_slope_4=5000
data_min_4=0
data_max_4=5
data_units_4="m^-1"
; ------------------------------------------------
data_ID_5=Lamp
```

```
data_database_name_5=Lamp
data_Channel_5=4
data_zero_offset_5=0
data_slope_5=1.0
data_min_5=0
data_max_5=5
data_units_5="VDC"

[Nephelometer]
Serial=0
data_count=5
; ----------first analog data point----------
; Channel_n is physical input channel
; zero_offset_n default is 0.0
; slope_n default is 1.0
; min_n is -10.0V
; max_n is +10.0V
data_ID_1=RH
data_database_name_1=RH
data_Channel_1=0
data_zero_offset_1=0.0
data_slope_1=20.0
data_min_1=0
data_max_1=5
data_units_1="%"
; ------------------------------------------------
data_ID_2=Scat Coef 1
data_database_name_2=Scat1
data_Channel_2=1
data_zero_offset_2=0
data_slope_2=25000
data_min_2=0
data_max_2=0.5
data_units_2="m^-1"
; ------------------------------------------------
data_ID_3=Scat Coef 2
data_database_name_3=Scat2
data_Channel_3=2
data_zero_offset_3=0
data_slope_3=5000
data_min_3=0
data_max_3=5
data_units_3="m^-1"
; ------------------------------------------------
data_ID_4=Calib
data_database_name_4=calib
```

```
data_Channel_4=3
data_zero_offset_4=0
data_slope_4=5000
data_min_4=0
data_max_4=5
data_units_4="m^-1"
; ------------------------------------------------
data_ID_5=Lamp
data_database_name_5=Lamp
data_Channel_5=4
data_zero_offset_5=0
data_slope_5=1.0
data_min_5=0
data_max_5=5
data_units_5="VDC"

[OZONE]
; Serial -- non-zero value means it is a serial instrument.
Serial=1
Port=6

OP_Code_Mask=0022
OP_Code_1=0022
parse_delimiters=" "
parse_endofrecord="\\n"
parse_recordsize=

data_count=1
data_ID_1=O3
data_database_name_1=O3
data_matchpattern_1="D "
data_offset_1=5
data_char_offset_1=0
data_zero_offset_1=0.0
data_slope_1=1.0


[Aethalometer]
Serial=1
Port=7
OP_Code_Mask=000F
OP_Code_1=0011

parse_delimiters=","
parse_endofrecord="\0D"
parse_recordsize=
```

```
data_count=8

data_ID_1=BC
data_database_name_1=BC
data_matchpattern_1="[0-9]+:[0-9]+"
data_offset_1=1
data_char_offset_1=0
data_zero_offset_1=0.0
data_slope_1=1.0

data_ID_2=Flow
data_database_name_2=FLOW
data_matchpattern_2="[0-9]+:[0-9]+"
data_offset_2=2

data_ID_3=sz
data_matchpattern_3="[0-9]+:[0-9]+"
data_offset_3=3

data_ID_4=sb
data_matchpattern_4="[0-9]+:[0-9]+"
data_offset_4=4

data_ID_5=rz
data_matchpattern_5="[0-9]+:[0-9]+"
data_offset_5=5

data_ID_6=rb
data_matchpattern_6="[0-9]+:[0-9]+"
data_offset_6=6

data_ID_7=frac
data_matchpattern_7="[0-9]+:[0-9]+"
data_offset_7=7

data_ID_8=atten
data_matchpattern_8="[0-9]+:[0-9]+"
data_offset_8=8

[BAM]
Serial=1
Port=5
OP_Code_Mask=000F
OP_Code_1=0011
Init_string="3"
```

parse_delimiters=", "
parse_endofrecord="*"
Recur_Cmd="3"
Recur_Freq=5

data_count=1
data_ID_1=Conc
data_units_1="mg"
data_database_name_1=Conc
data_matchpattern_1="=*[0-9]+[0-9]+:+[0-9]+[0-9]+"
data_offset_1=2
data_char_offset_1=0
data_zero_offset_1=0.0
data_slope_1=1.0

[Calibrator]
Serial=1
Port=9
parse_delimiters="\02\03"
parse_endofrecord="\03"
Recur_Cmd="\02FLOW 1 ACTUAL ?\03"
Recur_Freq=1
data_count=1
data_ID_1=Flow1
data_matchpattern_1="\06"
data_offset_1=0
data_char_offset_1=1

[NOY]
Serial=1
Port=8
parse_delimiters="\A0 "
parse_endofrecord="\0A"
Recur_Cmd="\A0no\0D"
Recur_Freq=1

data_count=2

data_ID_1=NOY
data_matchpattern_1="no "
data_offset_1=1
data_units_1="ppb"

data_ID_2=sum
data_matchpattern_2="sum "

```
data_offset_2=1

[PMS]
Serial=1
Port=10
init_string="TI 010\0D\0A"
parse_delimiters=" \0A"
parse_endofrecord="\03"
Recur_Cmd="SS\0D\0A"
Recur_Freq=1
data_count=1
data_ID_1=PMS-1
data_matchpattern_1="[0-9]+[0-9]+:+[0-9]+[0-9]+"
data_offset_1=18

[OCEC]
Serial=1
Port=12
parse_delimiters=",\0D"
parse_endofrecord="\0A"
data_count=1
data_ID_1=mc3
data_matchpattern_1="[0-9]+[0-9]+:+[0-9]+[0-9]+"
data_offset_1=4

[Climet]
; NOTES: no null modem required.
Serial=1
Port=11
init_string="X0010\0D"
parse_delimiters=",\0A\0D"
parse_endofrecord="[~S][\0A\0D]"
Recur_Cmd="S\0D"
Recur_Freq=1
data_count=18

data_ID_1=Flow
data_matchpattern_1="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_1=4
data_units_1=LPM

data_ID_2=Temp
data_matchpattern_2="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_2=5
data_units_2=F
```

```
data_ID_3=n1
data_matchpattern_3="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_3=6
data_units_3=

data_ID_4=n2
data_matchpattern_4="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_4=7
data_units_4=

data_ID_5=n3
data_matchpattern_5="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_5=8
data_units_5=

data_ID_6=n4
data_matchpattern_6="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_6=9
data_units_6=

data_ID_7=n5
data_matchpattern_7="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_7=10
data_units_7=

data_ID_8=n6
data_matchpattern_8="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_8=11
data_units_8=

data_ID_9=n7
data_matchpattern_9="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_9=12
data_units_9=

data_ID_10=n8
data_matchpattern_10="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_10=13
data_units_10=

data_ID_11=n9
data_matchpattern_11="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_11=14
data_units_11=

data_ID_12=n10
```

```
data_matchpattern_12="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_12=15
data_units_12=

data_ID_13=n11
data_matchpattern_13="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_13=16
data_units_13=

data_ID_14=n12
data_matchpattern_14="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_14=17
data_units_14=

data_ID_15=n13
data_matchpattern_15="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_15=18
data_units_15=

data_ID_16=n14
data_matchpattern_16="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_16=19
data_units_16=

data_ID_17=n15
data_matchpattern_17="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_17=20
data_units_17=

data_ID_18=n16
data_matchpattern_18="[0-9]+[0-9]+:[0-9]+[0-9]+"
data_offset_18=21
data_units_18=


; -------------------------------MATCHPATTERN Info:--------------------------------
.       Matches any character.
?       Matches zero or one instances of the expression preceding ?.
\       Cancels the interpretation of special characters
    (for example, \? matches a question mark). You can also use the following
    constructions for the space and nondisplayable characters:
            \b      backspace
            \f      form feed
            \n      newline
            \s      space
            \r      carriage return
```

\xx any character, where xx is the hex ascii code
\t tab

^ If ^ is the first character of regular expression, it anchors the match to the offset in string. The match fails unless regular expression matches that topic of string that begins with the character at offset. If ^ is not the first character, it is treated as a regular character.

[ ] Encloses alternates. For example, [abc] matches a, b, or c. The following character has special significance when used within the brackets in the following manner.

- (dash) Indicates a range when used between digits, or lowercase or uppercase letters (for example, [0-5],[a-g], or [L-Q])

The following characters have significance only when they are the first character within the brackets.

~ Excludes the set of characters, including nondisplayable characters. [~0-9] matches any character other than 0 through 9.

^ Excludes the set with respect to all the displayable characters (and the space characters). [^0-9] gives the space characters and all displayable characters except 0 through 9.

+ Matches the longest number of instances of the expression preceding +; there must be at least one instance to constitute a match.

* Matches the longest number of instances of the expression preceding * in regular expression, including zero instances.

$ If $ is the last character of regular expression, it anchors the match to the last element of string. The match fails unless regular expression matches up to and including the last character in the string. If $ is not last, it is treated as a regular character.

Calibration command files (.cal) located in D:\Calibrate have the same format as .ini-type files and define a sequence of calibration steps for the DAP to follow. These files are not accessed by the DAP until they are executed, so .cal files may be created, deleted, and modified during normal data acquisition. Calibration files may be executed on a regularly schedule by inclusion in the [Autocal] section of CRPAQS.INI. Calibration files may also be executed manually through the display panel. Only one calibration file may be executing at any time. A display panel switch allows a user to stop a calibration procedure that is in progress.

Calibration procedures are used primarily to switch valves controlling instrument inlet flows to ambient or calibrator sources, and to operate calibration instruments and pumps. As part of these procedures, serial commands may be sent to instruments, especially the calibrator. An example calibration file is included as Table 4.

Table 4. Example calibration command file (.cal).

```
 ; Any line starting with a number is parsed as
; a script command. All other lines ignored.
; The number represents the minutes past
; start time to execute the command.
; Strings to serial instruments must be contained in "".
; The second-last number is the serial port to
; receive the string command.
; The last number represents the control setting
; code (in “reverse binary”) to switch valves, run pumps, etc.
; Empty fields still must have commas.
;   ex. 20="",,0   will send 0 to digital outputs
; Fields must be seperated by commas.
[CAL_SCRIPT]
0="\002FLOW 1 TARGET=5000.0\003\002FLOW OZONE
TARGET=0.0\003\002FLOW UPDATE\003",9,01
5="",9,11
15="\002FLOW 1 ACTUAL ?\003\002FLOW OZONE ACTUAL ?\003",9,11
20="\002FLOW OZONE TARGET=0.08\003\002FLOW UPDATE\003",9,11
25="\002FLOW 1 ACTUAL ?\003\002FLOW OZONE ACTUAL ?\003",9,11
32="\002FLOW OZONE TARGET=0.0\003\002FLOW UPDATE\003",9,01
40="\002FLOW 2 TARGET=0.0\003\002FLOW 1 TARGET=0.0\003\002FLOW
UPDATE\003",9,01
41="",9,0
```